PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-069418
(43)Date of publication of application : 12.03.1996

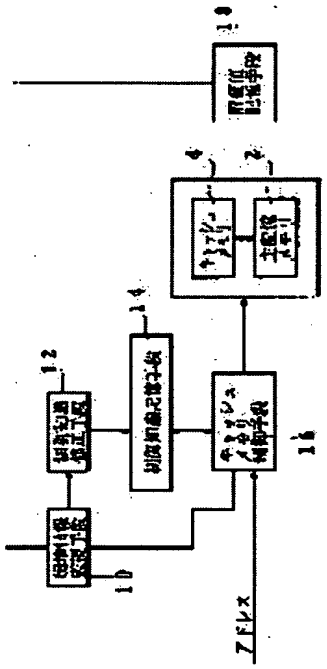
(51)Int.Cl. G06F 12/12
G06F 9/45
G06F 9/44
G06F 9/46
G06F 15/163

(21)Application number : 06-202565
(22)Date of filing : 26.08.1994

(71)Applicant : OMRON CORP
(72)Inventor : MAEDA TADASHI
SOGO TAIJI
TSUTSUMI YASUHIRO
TAJIMA TOSHIHIRO
ARAO MAKI
OYAGI MASAYUKI

(54) CACHE MEMORY CONTROL SYSTEM, COMILTER CONTROL SYSTEM, AND COMPUTER SYSTEM

(57)Abstract:
PURPOSE: To speed up and optimize the control over a computer device and its peripheral device.
CONSTITUTION: A cache memory control means 16 receives an address, and replaces the address of a cache memory 4 based on a control knowledge unless the cache memory 4 has the received address. A control knowledge storage means 14 is stored with the control knowledge. An environment information monitor means 10 obtains state information on access to the cache memory 4 from a cache memory control means 16 and calculates feature quantities including an access frequency and a cache mishit rate. Further, it is judged whether the cache mishit rate is higher than an evaluated value and when the cache mishit rate is higher, the feature quantities are outputted to a control knowledge correcting means 12, which is placed in operation. The control knowledge correcting means 12 corrects the control knowledge stored in the control knowledge storage means 14 based on the feature quantities.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平8－69418

(43)公開日　平成８年(1996)３月12日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	12/12		Ｆ　7623－5B				
	9/45						C4
	9/44	５５４　Ｚ	7737－5B				
			7737－5B	Ｇ０６Ｆ	9/44	３２２　Ｈ	C4
					15/16	３２０　Ｋ	

審査請求　未請求　請求項の数７　ＯＬ　（全 39 頁）　最終頁に続く

(21)出願番号　特願平6－202565

(22)出願日　平成６年(1994)８月26日

(71)出願人　000002945
オムロン株式会社
京都府京都市右京区花園土堂町10番地

(72)発明者　前田　匡
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

(72)発明者　十河　太治
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

(72)発明者　堤　康弘
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

(74)代理人　弁理士　古谷　栄男　（外２名）

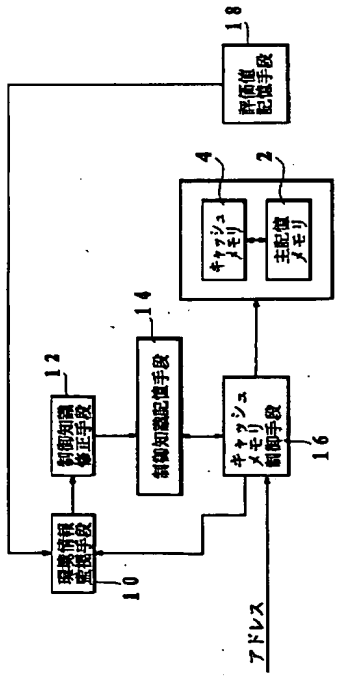
最終頁に続く

(54)【発明の名称】　キャッシュメモリ制御システム、コンパイラ制御システムおよびコンピュータ・システム

(57)【要約】

【目的】　コンピュータ装置およびその周辺装置の制御の高速化、最適化を目的とする。

【構成】　キャッシュメモリ制御手段１６は、アドレスを受けて、当該アドレスがキャッシュメモリになければ、制御知識に基づいて、キャッシュメモリのアドレスの置き換えを行う。制御知識記憶手段１４には、前記制御知識が記憶されている。環境情報監視手段１０は、キャッシュメモリ制御手段１６から、キャッシュメモリのアクセスの状態情報を得て、アクセス回数、キャッシュミス率を含む特徴量を算出する。さらに、キャッシュミス率が評価値よりも高いか否かを判断し、高い場合には、当該特徴量を制御知識修正手段１２に出力するとともに制御知識修正手段１２を動作させる。制御知識修正手段１２は、前記特徴量に基づいて、制御知識記憶手段１４に記憶された制御知識を修正する。



3

先度を決定し、
評価関数がタスク分散であるかタスク集中であるか、および評価値を記憶すしておき、
前記環境情報を得て、タスク分散度またはタスク集中度を算出するとともに、評価関数がタスク分散またはタスク集中のいずれであるかを判断し、
評価関数がタスク分散である場合には、タスク分散度が評価値を下回っている場合に、評価関数がタスク集中である場合には、タスク集中度が評価値を上回っている場合に、前記環境情報に基づいて、各タスクの重要度、混雑度を算出し、算出したタスクの重要度、混雑度に基づいて、前記タスク制御知識を修正するとともに、前記タスクの重要度、混雑度に基づいて、前記評価関数および評価値を修正すること、
を特徴とするコンピュータの動作制御方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】この発明は、コンピュータ装置およびその周辺装置の制御に関し、特にその最適化に関するものである。
【０００２】
【従来の技術】図６３に、従来のキャッシュメモリ制御システムの構成を示す。容量が大きくアクセスの遅い主記憶メモリ２と容量は小さいがアクセスの早いキャッシュメモリ４が設けられている。ＣＰＵ（図示せず）等からアドレスが与えられると、主記憶メモリ２の当該アドレスにアクセスするとともに、当該アドレスを含む1群（データブロックという）のデータをキャッシュメモリ４にコピーする。以後、このアドレスが与えられると、キャッシュメモリ制御手段6は、キャッシュメモリ４にアクセスを行う。したがって、アクセススピードが向上する。
【０００３】図６４に示すように、キャッシュメモリ４に、Ｃ１～Ｃ４の4つのデータブロックが設けられており、それぞれにＢ３、Ｂ６、Ｂ９、Ｂ１４がコピーされているとする。なお、図中の「０」～「６０」の数字はアドレスを示している。この状態において、アドレス「４」が与えられると、キャッシュメモリ制御手段6は、キャッシュメモリ４中の何れかのデータブロックを削除する。その後、アドレス「４」の属するデータブロック「Ｂ２」を削除したデータブロックの代りにコピーする。このようにして、キャッシュメモリ４の記憶内容は、次々と新しいデータブロックに置き換えられていく。
【０００４】ここで、キャッシュメモリ４のどのデータブロックを削除するかは、キャッシュメモリ制御手段6が、図６５の手順によって決定していた。まず、ＬＲＵフラグを記憶するためのカウンタバッファを設ける。ここで、ＬＲＵフラグとは、各データブロックについて、最後に使用したのが現在から何回前であったかを現わす

4

フラグである。ここでは、「０」が現在使用中、「１」が1回前に使用、「２」が2回前に使用、「３」が3回以前に使用したことを現わすものとする。
【０００５】キャッシュメモリが図６４の状態にあるとき、与えられたアドレスが「９」であれば、キャッシュメモリ制御手段6は、キャッシュメモリ４のデータブロックＣ１にアクセスする（図６５参照）。この時、ブロックＣ１のＬＲＵフラグを「０」にする。他のブロックＣ２～Ｃ４のＬＲＵフラグは「３」のままである。
【０００６】次に与えられたアドレスが「２０」であれば、キャッシュメモリ４のデータブロックＣ４にアクセスする。この時、ブロックＣ４のＬＲＵフラグを「０」にするとともに、ブロックＣ１のブロックＣ１のＬＲＵフラグを「１」に変える。他のブロックＣ２、Ｃ３のＬＲＵフラグは「３」のままである。
【０００７】その後、アドレスが「５３」「３２」「１０」「２１」と与えられると、各ブロックＣ１、Ｃ２、Ｃ３、Ｃ４のＬＲＵフラグは、図に示すように「１」「２」「３」「０」となる。次に、アドレス「１２」が与えられたとする。この場合、キャッシュメモリ４にはこのアドレスがコピーされていないので（キャッシュミスという）、主記憶メモリ２にアクセスする。これとともに、アドレス「１２」を含むデータブロックＢ４をキャッシュメモリ４にコピーする。つまり、キャッシュメモリ４のいずれかのデータブロックを削除して、データブロックＢ４に置き換える、この際、キャッシュメモリ４の置き換えは、ＬＲＵフラグの最も大きいデータブロック、すなわち、最も古く使われたデータブロックが対象となる。したがって、この場合には、データブロックＣ３が置き換え対象となる。
【０００８】このようにして置き換えられたキャッシュメモリ４の内容を示す。ブロックＣ３にデータブロックＢ４の内容がコピーされている。以上のようにして、キャッシュメモリ４の記憶内容は、次々と新しいデータブロックに置き換えられていく。
【０００９】図６７に、従来のコンパイル装置の構成を示す。高級言語によるソースプログラムが与えられると、コンパイラ１００は、これを実行形式のプログラムに変換する。この際、ソースプログラム中の変数に対して、メモリまたはレジスタを割り当てる。メモリよりもレジスタの方が高速であるから、出現頻度の高い変数に対して、レジスタを割り当てるようにしている（制御知識）。
【００１０】図６８に、従来のコンパイル装置におけるレジスタ割り当てのフローチャートを示す。まず、ステップＳ１０２において、ソースコードの対象行に変数が存在するか否かを判定する。存在しなければ、ステップＳ１０６に進む。存在する場合には、当該変数がすでに抽出されている変数か否かを判定する（ステップＳ１０３）。抽出されている変数の場合には、当該変数の出現

7

【００２３】この発明は、上記のような問題点を解決して、状況に応じて適切なレジスタ割り当てを得ることのできるコンパイル制御システムおよび変数割り当て方法を提供することを目的とする。

【００２４】また、図７０、図７１に示すようなコンピュータ・システムにおいては、次のような問題点があった。従来のシステムでは、各ＣＰＵの負荷の均等化を図って、特定のＣＰＵに負担がかからないようにし、タスク全体の処理速度を維持するようにしている。しかし、全タスク量が少ない場合であっても、タスクの分散を図るように制御していた。このような場合には、タスクが分散されているため、ネットワークを介しての通信時間が大きくなり、かえって、処理速度が遅くなるという問題があった。

【００２５】この発明は、上記のような問題点を解決して、状況に応じて適切なタスク制御を行うことのできるシステムおよび方法を提供することを目的とする。

【００２６】

【課題を解決するための手段】請求項１、２のキャッシュメモリ制御システムは、アドレスを受けて、当該アドレスがキャッシュメモリになければ、制御知識に基づいて、キャッシュメモリのアドレスの置き換えを行うキャッシュメモリ制御手段、前記制御知識を記憶する制御知識記憶手段、キャッシュメモリ制御手段から、キャッシュメモリのアクセスの状態情報を得て、アクセス回数、キャッシュミス率を含む特徴量を算出し、キャッシュミス率が評価値よりも高いか否かを判断し、高い場合には、当該特徴量を制御知識修正手段に出力するとともに制御知識修正手段を動作させる環境情報監視手段、前記特徴量に基づいて、制御知識記憶手段に記憶された制御知識を修正する制御知識修正手段、を備えている。

【００２７】ここで、環境情報監視手段は、実施例においては、図３、図４に示すフローチャートが対応する。また、制御知識修正手段は、実施例においては、図４のステップＳ１２が対応する。なお、制御知識修正手段は、実施例では、図７のルール、図８のメンバーシップ関数に基づいて、ファジイ推論を行って制御知識の修正をしているが、環境情報監視手段からの特徴量に基づき、ファジイ推論に拘らず、制御知識の修正を行うものを含むものである。制御知識記憶手段に記憶される制御知識は、実施例では、図９、図１０に示すものが対応する。なお、制御知識とは、キャッシュメモリのアドレス置き換えを行う際に、用いる規則、原則、優先度等をいうものである。キャッシュメモリ制御手段は、実施例においては、図１１のフローチャートが対応する。

【００２８】請求項３のキャッシュメモリ制御方法は、キャッシュメモリへのアクセスにおけるキャッシュミス率を計測し、キャッシュミス率が所定条件よりも悪い場合には、前記制御知識を変更することを特徴としている。

8

【００２９】請求項４のコンパイラ制御システムは、ソース形式のプログラムを受けて、実行形式のプログラムを出力するとともに、制御知識に基づいて各変数をレジスタもしくはメモリに割り当てるコンパイラ制御手段、前記制御知識を記憶する制御知識記憶手段、コンパイラ制御手段から、変数情報およびループプログラム情報を得るとともに、コンパイラ評価手段からの評価値が所定の値よりも小さい場合には、前記変数情報およびループプログラム情報に基づいて変数出現度、ループプログラム出現度を含む特徴量を算出する環境情報監視手段、前記特徴量に基づいて、制御知識記憶手段に記憶された制御知識を修正する制御知識修正手段、を備えている。

【００３０】ここで、コンパイラ制御手段は、実施例においては、図１５のステップＳ１２０が対応する。環境情報監視手段は、実施例においては、図１５のステップＳ１２２以下、および図１６のフローチャートが対応する。また、変数情報とは、特徴量を得るために必要な変数に関する情報をいうものであり、実施例では、変数の出現回数等がこれに対応する。環境変化理解手段は、実施例においては、図１８のフローチャートが対応する。制御知識修正手段は、実施例においては、図２３のフローチャートが対応する。制御知識修正手段は、実施例では、図２４のルール、図２５のメンバーシップ関数に基づいて、ファジイ推論を行って制御知識の修正をしているが、環境情報監視手段からの特徴量に基づき、ファジイ推論に拘らず、制御知識の修正を行うものを含むものである。

【００３１】請求項５のコンパイラにおける変数のレジスタへの割り当て制御方法は、変数をレジスタに割り当てるための制御知識を記憶しておき、コンパイル対象の高級言語ソースプログラムの中での各変数の出現状況を示す情報を抽出し、前記制御知識と出現状況に基づいて、レジスタに割り当てるべき変数を決定し、決定したレジスタ割り当てもとで、前記高級言語ソースプログラムをコンパイルした後に実行形式プログラムを生成し、前記実行形式プログラムの実行速度が所定の基準値よりも小さい場合には、前記制御知識を修正した後に、前記高級言語ソースプログラムを再度コンパイルすることを特徴としている。

【００３２】請求項６のコンピュータ・システムは、各ＣＰＵの負荷状況、主記憶装置の残り容量、既に実行されているタスクの処理状況、実行予定のタスクを含む環境情報を受けて、タスク制御知識に基づいて、各タスクをどのＣＰＵで処理するか、および各タスクの優先度を決定するタスク制御手段、前記タスク制御知識を記憶するタスク制御知識記憶手段、評価関数がタスク分散であるかタスク集中であるかを記憶するとともに、評価値を記憶するタスク評価値記憶手段、タスク制御手段から、前記環境情報を得て、タスク分散度またはタスク集中度を算出するとともに、タスク評価値記憶手段に記憶され

ち、最新の１０２４回分がレジスタに記憶される。

【００４０】次に、ステップＳ３において、キャッシュミスが発生したか否かを判定する。キャッシュミスが発生していなければ、ステップＳ１に戻る。キャッシュミスが発生すれば、キャッシュミス記憶用のレジスタに、各ブロックＣ１～Ｃ４ごとに記録する（ステップＳ４）。このレジスタの記憶状態は、例えば、図２Ｂのようになる。図中、「１」の部分は、アクセスミスがあったことを示している。

【００４１】次に、リード・ライト(Read-Write)が発生したかどうかを判定する（ステップＳ５）。ここで、リード・ライト(Read-Write)とは、書込を行った場合をいう。なお、アクセスとは、読み出しだけを行った場合も含む。リード・ライト(Read-Write)が発生していなければ、ステップＳ７に進む。発生していれば、図2と同様の構成のリード・ライト(Read-Write)記憶用レジスタに記憶する（ステップＳ６）。

【００４２】次に、アクセス記憶用レジスタとキャッシュミス記憶用のレジスタに基づいて、キャッシュメモリ全体（全ブロック）のキャッシュミス率を演算する（ステップＳ７）。このキャッシュミス率が、評価値記憶手段１８に記憶された評価値（この実施例では１０％）より小さい場合には、ステップＳ１に戻る。つまり、制御知識を修正する必要はないと判断し、現在の制御知識のままで制御を進める。

【００４３】一方、キャッシュミス率が、１０％より大きい場合には、ステップＳ９に進む。ステップＳ９においては、各ブロックごとのキャッシュミス発生率を演算する。次に、各ブロックごとのリード・ライト(Read-Write)発生率、全体のリード・ライト(Read-Write)発生率を算出する（ステップＳ１０、Ｓ１１）。このようにして算出したキャッシュミス発生率、リード・ライト(Read-Write)発生率を、特徴量として制御知識修正手段１２に渡す。例えば、各記憶用レジスタの集計内容が図5に示すような状態であった場合には、図6に示すような特徴量が出力される。

【００４４】制御知識修正手段１２は、この特徴量に基づいて制御知識の修正を行う（ステップＳ１２）。この実施例では、制御情報として、ＬＲＵ情報だけを用いるもの、ＬＲＵ情報と使用頻度情報とを用いるもの、ＬＲＵ情報とＲｅａｄ－Ｗｒｉｔｅ情報とを用いるもの、ＬＲＵ情報と使用頻度情報とＲｅａｄ－Ｗｒｉｔｅ情報とを用いるものの4つの内から選ぶようにしている。

【００４５】制御知識修正手段１２は、特徴量に基づいて、図7のルールおよび図8のメンバーシップ関数を用いて、各ブロックごとにファジイ推論を行う。この実施例では、後件部の適合度が０．９を超えるものが1ブロックでも存在する場合には、当該後件部にしたがって、制御知識を変更（知識の追加、削除）する。例えば、現在、ＬＲＵ情報だけを制御情報として用いているとして、「使用頻度情報を制御知識に加える」という後件部の適合度が０．９を超えた場合には、ＬＲＵ情報と使用頻度情報とを用いるものに制御情報を変更する。

【００４６】この実施例において用いた4つの制御情報を、図9、図１０に示す。制御知識記憶手段１４には、制御知識修正手段１２によって選択された制御知識が記憶される。キャッシュメモリ制御手段１６は、この制御知識に基づいて制御、すなわち、キャッシュミスが発生した場合にどのブロックを置き換えるかの制御を行う。なお、この実施例では、上記置き換え制御のために、図１２に示すような、ＬＲＵフラグ、使用頻度フラグ、Ｒｅａｄ－Ｗｒｉｔｅフラグを設けている。

【００４７】ＬＲＵフラグは、各ブロックＣ１～Ｃ４に対して、今回アクセスした場合には「０」、前回アクセスした場合には「１」、前々回アクセスした場合には「２」、３回前にアクセスした場合には「３」が与えられる。図１２Ａは、ＬＲＵフラグの時間的変化を示している。使用頻度フラグは、図2Ａのレジスタと同じものである。また、Ｒｒａｄ－Ｗｒｉｔｅフラグは、図2Ｂのレジスタと同じものである。

【００４８】キャッシュミスが発生すると、キャッシュメモリ制御手段１６は、図１１の置き換えブロック決定処理を行う。ここでは、図１０Ｂの制御知識（ＬＲＵ情報と使用頻度情報とＲｅａｄ－Ｗｒｉｔｅ情報とを用いるもの）が用いられているものとして説明を行う。

【００４９】まず、各ブロックＣ１～Ｃ４ごとに、ＬＲＵフラグの重み係数を決定する（ステップＳ２０）。つまり、図１０Ｂの制御知識に基づき、ＬＲＵフラグの値に対応した重み係数を算出する。この例でいえば、ブロックＣ１のＬＲＵフラグは「３」であるから、重み係数は「１．０」となる（図１３参照）。同様に、ブロックＣ２の重み係数は「０．５」、ブロックＣ３の重み係数は「０．７５」、ブロックＣ４の重み係数は「０．２５」となる。

【００５０】次に、各ブロックＣ１～Ｃ４ごとに、使用頻度フラグの重み係数を決定する（ステップＳ２１）。この重み係数を得るため、まず、図１２Ｂに示すように、各ブロックごとのアクセス回数を計数する。このアクセス回数に基づき、図１０Ｂの制御知識を使って、使用頻度フラグの重み係数を算出する。この例でいえば、ブロックＣ１のアクセス回数は他のどのブロックよりも多いので、重み係数は「０．２５」となる（図１３参照）。同様に、ブロックＣ２の重み係数は「０．７５」、ブロックＣ３の重み係数は「０．５」、ブロックＣ４の重み係数は「０．５」となる。

【００５１】次に、各ブロックＣ１～Ｃ４ごとに、Ｒｅａｄ－Ｗｒｉｔｅフラグの重み係数を決定する（ステップＳ２２）。この重み係数を得るため、まず、図１２Ｃに示すように、各ブロックごとのＲｅａｄ－Ｗｒｉｔｅ回数を計数する。このアクセス回数に基づき、図１０Ｂ

15

を求める（ステップＳ１５６）。たとえば、図１９Ｃに示すようになる。

【００６４】次に、各サイクル時間における、各変数に対するレジスタの割り当て状況を取得する（ステップＳ１５８）。

【００６５】制御知識修正手段１０６は、環境情報監視手段１１０と環境変化理解手段１０８からの出力に基づいて、制御知識の修正を行う。この実施例では、制御情報として、出現頻度のみを用いるもの、出現頻度と移動平均とを用いるもの、出現頻度とループプログラム出現頻度とを用いるもの、出現頻度と移動平均とループプログラム出現頻度とを用いるものの4つの中から選ぶようにしている。

【００６６】制御知識修正手段１０６は、図２３のフローチャートにしたがって、制御知識の修正を行う。まず、各変数ごとに、各サイクル時間ごとに、制御知識候補の推論を行う（ステップＳ１６０）。なお、推論は、ステップＳ１５０、１５４、１５６で求めた、出現場所（出現頻度がしきい値より大きいサイクル時間）についてのみ行う。この推論は、図２４のルール、図２５のメンバーシップ関数に基づいて行う。後件部の適合度が０．９を超える制御知識候補がある場合には、当該後件部にしたがって、制御知識を変更する（ステップＳ１６４、１６６）。たとえば、現在、出現頻度だけを制御情報として用いているとして、「移動平均情報を制御知識に加える」という後件部の適合度が０．９を超えた場合には、出現頻度と移動平均とを用いるものに制御情報を変更する。

【００６７】制御知識記憶手段１０４には、制御知識修正手段１０６によって選択された制御知識が記憶される。コンパイラ制御手段１０２は、この制御知識に基づいて制御、すなわちどの変数をレジスタに割り当てるのかの制御を行う。なお、この実施例では、上記割り当て制御のために、図２６に示すような、出現頻度カウンタバッファ、移動平均バッファ、ループプログラム出現頻度カウンタを設けている。

【００６８】出現頻度カウンタバッファは、コンパイル対象となっている当該サイクル時間における変数の出現頻度をカウントするものである。また、移動平均バッファは、コンパイル対象となっているサイクル時間、１つ前のサイクル時間および2つ前のサイクル時間の出現頻度を平均した値を保持するためのバッファである。ループプログラム出現頻度カウンタバッファは、コンパイル対象となっている当該サイクル時間におけるループプログラム中の変数の出現頻度をカウントするものである。

【００６９】図２７に、コンパイラ制御手段１０２のフローチャートを示す。まず、ソースプログラムの対象となるステップに変数が存在するか否かを判断する（ステップＳ１７０）。存在しなければ、ステップＳ１７４に進む。存在すれば、当該変数がすでに抽出されている変数であるか否かを判断する（ステップＳ１７１）。すでに抽出されている変数であれば、当該変数の出現頻度カウンタをインクリメントする（ステップＳ１７２）。抽出されていない変数であれば、当該変数のための出現頻度カウンタを用意するとともに、カウント数を「１」とする（ステップＳ１７３）。

16

【００７０】次に、ステップＳ１７４において、ループプログラムが存在するか否かを判定する。存在しなければ、ステップＳ１８０に進む。存在すれば、当該ループプログラムの中に変数が存在するか否かを判断する（ステップＳ１７５）。存在しなければ、ステップＳ１８０に進む。存在すれば、当該変数がすでに抽出されている変数であるか否かを判断する（ステップＳ１７６）。すでに抽出されている変数であれば、当該変数のループプログラム出現頻度カウンタをインクリメントする（ステップＳ１７８）。抽出されていない変数であれば、当該変数のための出現頻度カウンタを用意するとともに、カウント数を「１」とする（ステップＳ１７９）。

【００７１】次に、ステップＳ１８０において、所定ステップ数のコンパイルが進んだかどうか（サイクル時間が経過したかどうか）を判定する。経過していなければ、ステップＳ１７０以下を繰り返し実行する。経過していれば、レジスタの割り当てを決定する（ステップＳ１８１）。

【００７２】レジスタの割り当て決定のフローチャートを、図３０、図３１に示す。なお、ここでは、図２２Ｂに示す制御知識を用いて割り当ての決定を行うものとして説明を行う。まず、各変数ごとに、出現頻度がしきい値α（この場合は0である（図２２Ｂ参照））より大きな変数に対し、係数を決定する（ステップＳ１８６）。例えば、図３２に示すように、各変数ｖａｌ１，ｖａｌ２，ｖａｌ３の出現頻度カウンタの値が、「１００」「１１０」「１２０」であったとすれば、それぞれの重み係数は、図２２Ｂより「０．５」「０．７５」「１．０」と決定される。次に、同様にして、ループプログラム出現頻度カウンタに基づき、その重み係数を決定する（ステップＳ１８８）。

【００７３】次に、１回前のサイクル時間の頻度カウンタの値、２回前のサイクル時間の頻度カウンタの値、現在のサイクル時間の頻度カウンタの値に基づいて、各変数ごとに、移動平均を算出する（ステップＳ１８８）。この移動平均に基づいて、図２２Ｂより、重み係数を算出する（ステップＳ１８９）。

【００７４】このようにして、各変数ごとに3つの重み係数を算出した後、この重み係数を各変数ごとに合計して、評価値とする（ステップＳ１９０）。次に、この評価値の上位3つまでの変数を選定する（ステップＳ１９１）。ここで、レジスタの数よりも選定された変数の数の方が多いか否かを判定する（ステップＳ１９２）。少ない場合には、選定された変数をれじすたに割り当てる

スク集中度がタスク評価値記憶手段２１４に記憶された評価値（ここでは２０％以下とする）以下であるか否かを判断する。ここで、各ＣＰＵごとのタスク集中度が全て２０％以下である場合のみ、タスク集中度が２０％以下であるとする（図４０参照）。

【００８５】タスク集中度が２０％以下であれば、ステップＳ３１０に戻る。タスク集中度が２０％以下でなければ、ステップＳ３２２以下を実行し、環境変化理解手段２２０を動作させる。

【００８６】以上のように、環境情報監視手段２２２は、環境情報から算出されたタスク分散度またはタスク集中度が、記憶された評価値に適合していない場合に、環境変化理解手段２２０を動作させる。

【００８７】図４１に、環境変化理解手段２２０のフローチャートを示す。環境変化理解手段２２０は、環境情報監視手段２２２からの出力を受けて動作する。まず、ステップＳ３３０において、各タスクの重要度を算出する。この実施例では、各タスクの発生頻度、各タスクの処理平均時間に基づいて、タスクの重要度をファジイ推論する。つまり、図４９Ａ，Ｂの前件部のメンバーシップ関数、図４８のルール、図４９Ｃの後件部のメンバーシップ関数に基づいて、タスクの重要度を求める。その結果を、図５０に示すタスクの重要度バッファに記憶する（ステップＳ３３１）。

【００８８】次に、分散コンピューティング環境の混雑度を、各ＣＰＵの負荷、全体のタスク数に基づいて、ファジイ推論する（ステップＳ３３２）。つまり、図５２Ａ，Ｂの前件部のメンバーシップ関数、図５１のルール、図５２Ｃの後件部のメンバーシップ関数に基づいて、分散コンピューティング環境の混雑度を求める。ここで、分散コンピューティング環境の混雑度とは、ＣＰＵの負荷や、ネットワークの混み具合によって表される、システム全体の混雑度合いをいう。タスク重要度および分散コンピューティング環境の混雑度は、環境情報評価手段２１６および制御知識修正手段２１８に出力される。

【００８９】次に、図４２に、環境情報評価手段２１６のフローチャートを示す。まず、ステップＳ３４０において、タスクの頻度、分散コンピューティング環境の混雑度、タスク重要度に基づいて、評価関数をタスク分散とするか、タスク集中とするかを推論する。すなわち、図５４Ａ，Ｂ，Ｃの前件部のメンバーシップ関数、図５３のルール、図５４Ｄの後件部のメンバーシップ関数に基づいて、評価関数をタスク分散とするか、タスク集中とするかを各タスク毎に推論する。その結果、タスク分散とすべき推論値とタスク集中とすべき推論値の何れが大きいかを判断する（ステップＳ３４１）。

【００９０】タスク集中の方が大きい場合には、評価関数にタスク集中を採用することとし（ステップＳ３４２）、当該評価関数および評価値をタスク評価値記憶手段２１４に書き込む（ステップＳ３４３）。タスク集中の場合の、評価関数および評価値の一例を、図４３Ｂに示す。

【００９１】一方、タスク分散の方が大きい場合には、評価関数にタスク分散を採用することとし（ステップＳ３４４）、当該評価関数および評価値をタスク評価値記憶手段２１４に書き込む（ステップＳ３４５）。タスク分散の場合の、評価関数および評価値の一例を、図４３Ａに示す。

【００９２】このようにして、環境に応じて、適切な評価関数、評価値が選択される。

【００９３】また、環境変化理解手段２２０からの出力は、制御知識修正手段２１８にも与えられている。図４４に、制御知識修正手段２１８のフローチャートを示す。まず、ステップＳ３５０において、各タスク毎に、分散コンピューティング環境の混雑度、タスクの頻度、タスクの重要度、タスクの優先度に基づいて、どの制御知識を用いるかをファジイ推論する。すなわち、図５６Ａ，Ｂ，Ｃ，Ｄの前件部のメンバーシップ関数、図５５のルール、図５６Ｅの後件部のメンバーシップ関数に基づいて、その制御知識を用いるかを各タスク毎に推論する。

【００９４】次に、これら推論結果中に、適合度０．９以上のものがあるか否かを判断する（ステップＳ３５１）。あれば、当該適合度０．９以上の知識をタスク制御知識記憶手段２１２に書き込む。なお、適合度０．９以上のものが複数ある場合には、それら全ての知識を書き込む（ステップＳ３５２）。適合度０．９以上のものがなければ、知識の修正は行わない。

【００９５】図４５、図４６に、タスク制御知識の例を示す。図４５において、例えば、タスクを他のＣＰＵに分散してもよい場合には、全てのＣＰＵの処理許可フラグが「１」とされる。タスクを自分のマシンのＣＰＵのみで行う場合には、タスク要求が発生したマシンのＣＰＵの処理許可フラグが「１」とされ、他のマシンのＣＰＵの処理許可フラグが「０」とされる。何れの知識を採用するかは、上述の制御知識修正手段２１８によって決定される（図５６Ｅ参照）。

【００９６】図４６には、タスク優先順位を決定するための制御知識の例が示されている。図４６Ａが処理時間のみで優先順位を決定する知識であり、図４６Ｂが使用頻度も考慮して優先順位を決定する知識である。何れの知識を採用するかは、上述の制御知識修正手段２１８によって決定される（図５６Ｅ参照）。

【００９７】最後に、タスク制御手段２１０の動作を説明する。まず、タスク実行先決定のフローチャートを図５９に示す。まず、タスク実行先決定用フラグを読み込む（ステップＳ３６０）。つまり、制御知識によって決定された各ＣＰＵの処理許可フラグを読み込む。次に、ステップＳ３６１において、フラグが「０」であるＣＰ

23

【図２７】コンパイラ制御手段１０２の動作を示すフローチャートである。
【図２８】コンパイラ制御手段１０２の動作を示すフローチャートである。
【図２９】コンパイラ制御手段１０２の動作を示すフローチャートである。
【図３０】コンパイラ制御手段１０２の動作を示すフローチャートである。
【図３１】コンパイラ制御手段１０２の動作を示すフローチャートである。
【図３２】コンパイラ制御手段１０２のレジスタ割り当てを示す図である。。
【図３３】実施例におけるレジスタ割り当ての状態を示す図である。
【図３４】この発明の一実施例によるコンピュータ・システムを示す図である。
【図３５】環境情報監視手段２２２の動作を示すフローチャートである。
【図３６】環境情報監視手段２２２の動作を示すフローチャートである。
【図３７】環境情報監視手段２２２の動作を示すフローチャートである。
【図３８】環境情報監視手段２２２の動作を示すフローチャートである。
【図３９】ＣＰＵの負荷レジスタを示す図である。
【図４０】ＣＰＵ負荷バッファを示す図である。
【図４１】環境変化理解手段２２０の動作を示すフローチャートである。
【図４２】環境情報評価手段２１６の動作を示すフローチャートである。
【図４３】評価知識フラグと評価目標値を示す図である。
【図４４】制御知識修正手段２１８の動作を示すフローチャートである。
【図４５】タスク実行先決定用フラグの例を示す図である。
【図４６】タスク優先順位用重み係数の例を示す図である。
【図４７】環境情報の一例を示す図である。
【図４８】各タスクの重要度を決定するためのルールを示す図である。
【図４９】各タスクの重要度を決定するためのメンバーシップ関数を示す図である。
【図５０】タスクの重要度バッファを示す図である。
【図５１】分散コンピューティング環境の混雑度を決定するためのルールを示す図である。
【図５２】分散コンピューティング環境の混雑度を決定するためのメンバーシップ関数を示す図である。
【図５３】評価関数を決定するためのルールを示す図である。

24

【図５４】評価関数を決定するためのメンバーシップ関数を示す図である。
【図５５】制御知識を決定するためのルールを示す図である。
【図５６】制御知識を決定するためのメンバーシップ関数を示す図である。
【図５７】環境情報および使用頻度レジスタを示す図である。
【図５８】タスク実行先決定用フラグ、タスク優先順位用重み係数を示す図である。
【図５９】タスク実行先決定制御のフローチャートを示す図である。
【図６０】タスク実行先決定用フラグを示す図である。
【図６１】優先度決定処理のフローチャートを示す図である。
【図６２】優先度決定処理を説明するための図である。
【図６３】従来のキャッシュメモリ制御システムを示す図である。
【図６４】従来のキャッシュメモリの割り当てを示す図である。
【図６５】ＬＲＵフラグを示す図である。
【図６６】キャッシュメモリに割り当てられたブロックを示す図である。
【図６７】従来のコンパイラシステムを示す図である。
【図６８】従来のコンパイラシステムの動作を示すフローチャートである。
【図６９】従来のコンパイラシステムにおけるレジスタ割り当て状況を示す図である。
【図７０】従来のコンピュータ・システムのブロック図を示す図である。
【図７１】従来のコンピュータ・システムのブロック図を示す図である。
【図７２】従来のタスク制御処理を示す図である。
【図７３】従来のタスク制御知識を示すである。
【符号の説明】
１０・・・環境情報監視手段
１２・・・制御知識修正手段
１４・・・制御知識記憶手段
１６・・・キャッシュメモリ制御手段
１０２・・・コンパイラ制御手段
１０４・・・制御知識記憶手段
１０６・・・制御知識修正手段
１１０・・・環境情報監視手段
２１０・・・タスク制御手段
２１２・・・タスク制御知識記憶手段
２１４・・・タスク評価値記憶手段
２１６・・・環境情報評価手段
２１８・・・制御知識修正手段
２２０・・・環境変化理解手段
２２２・・・環境情報監視手段

【図2】

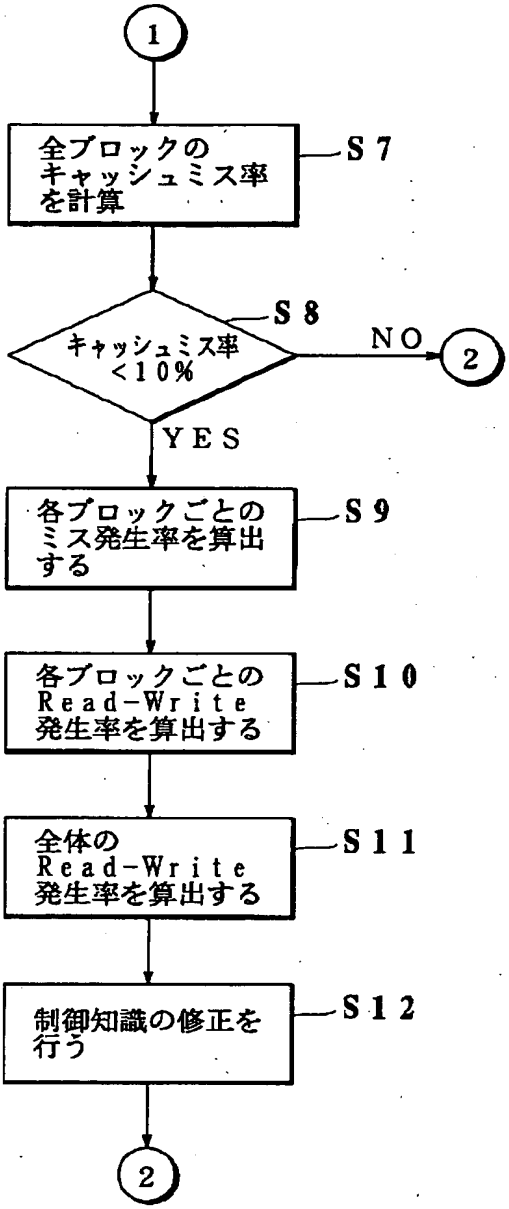
A

ブロックC1	1	1	0	0	1	1	0	0				0	1	1	0	1	1	1	1	0
ブロックC2	0	0	0	1	0	0	1	0				0	0	0	1	0	0	0	0	1
ブロックC3	0	0	1	0	0	0	0	1				0	0	0	0	0	0	0	0	0
ブロックC4	0	0	0	0	0	0	0	0				1	0	0	0	0	0	0	0	0
																	2回目	前回	今回	

B

ブロックC1	0	0	0	0	0	1	0	0				0	1	0	0	0	0	0	0	0
ブロックC2	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	1
ブロックC3	0	0	0	0	0	0	0	1				0	0	0	0	0	0	0	0	0
ブロックC4	0	0	0	0	0	0	0	0				1	0	0	0	0	0	0	0	0
																	2回目	前回	今回	

【図4】



【図5】

ブロック番号	最近1024回のキャッシュメモリアクセス回数	最近1024回のキャッシュメモリアクセス時のキャッシュミス発生回数	最近1024回のキャッシュメモリアクセス時のRead-Write回数
1	512	71	15
2	256	35	30
3	192	19	0
4	64	6	1
合計	1024	131	36

【図6】

ブロック番号	最近1024回のキャッシュメモリアクセス回数	最近1024回のキャッシュメモリアクセス時のキャッシュミス発生率	最近1024回のキャッシュメモリアクセス時のRead-Write発生率
1	512	14%	3%
2	256	14%	12%
3	192	10%	0%
4	64	9%	2%
合計	1024	13%	4%

【図7】

IF
使用頻度が高い、かつ、ミス発生率が高い

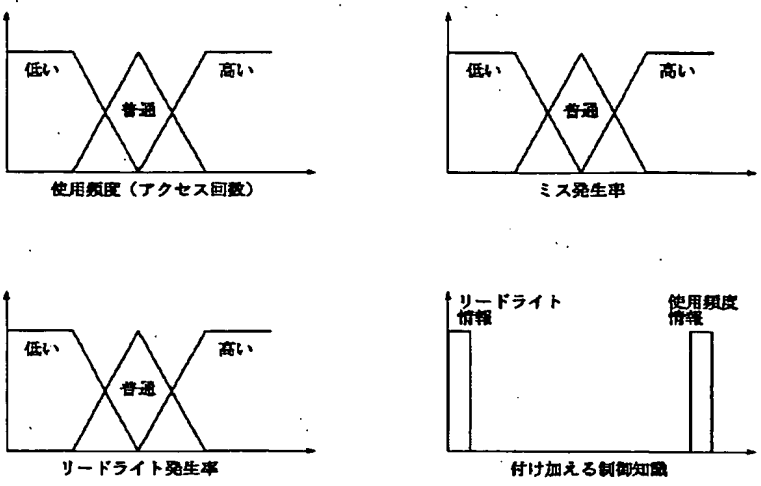
Then
使用頻度情報を制御知識に加える

IF
ミス発生率が高い、かつ、リード・ライト発生率が高い

Then
リード・ライト情報を制御知識に加える

・
・
・

【図8】



【図9】

A

・ＬＲＵ情報だけを用いる

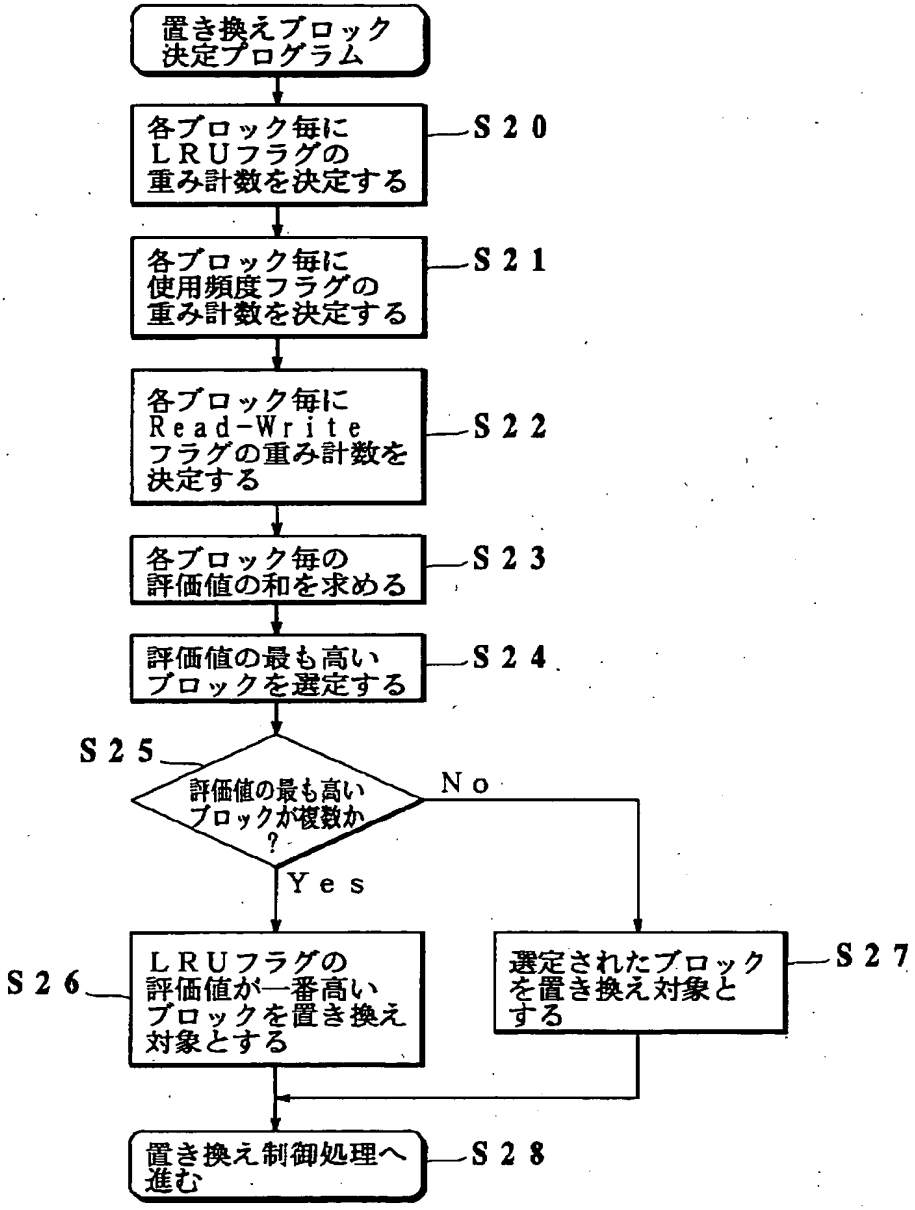
ＬＲＵフラグ	現在使用中	１つ前に使用	２つ前に使用	３つ前に使用
	0	0	0	1．0
使用頻度フラグ	使用頻度１位	使用頻度２位	使用頻度３位	使用頻度４位
	0	0	0	0
Read-Writeフラグ	Read-Write回数１位	Read-Write回数２位	Read-Write回数３位	Read-Write回数４位
	0	0	0	0

B

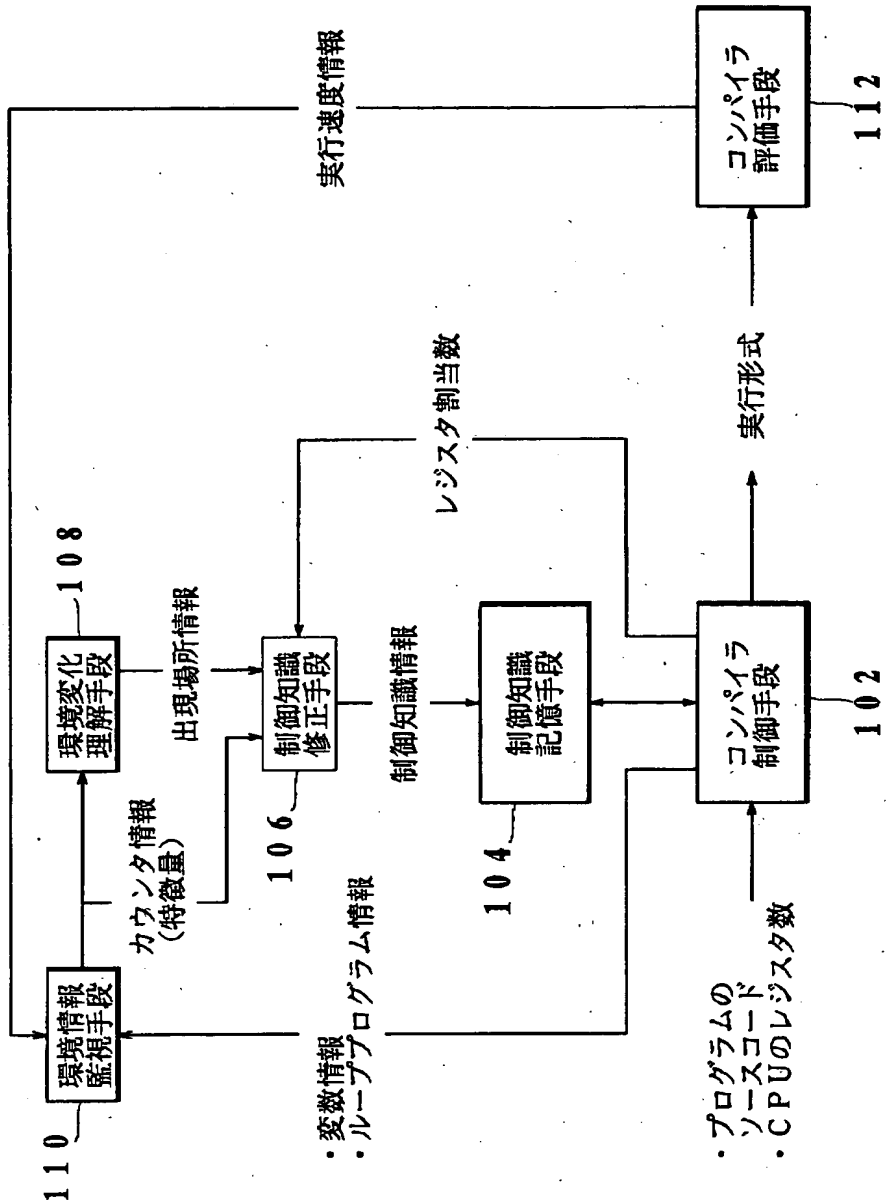
・ＬＲＵ情報と使用頻度とを用いる

ＬＲＵフラグ	現在使用中	１つ前に使用	２つ前に使用	３つ前に使用
	0．２５	0．５	0．７５	1．0
使用頻度フラグ	使用頻度１位	使用頻度２位	使用頻度３位	使用頻度４位
	0．２５	0．５	0．７５	1．0
Read-Writeフラグ	Read-Write回数１位	Read-Write回数２位	Read-Write回数３位	Read-Write回数４位
	0	0	0	0

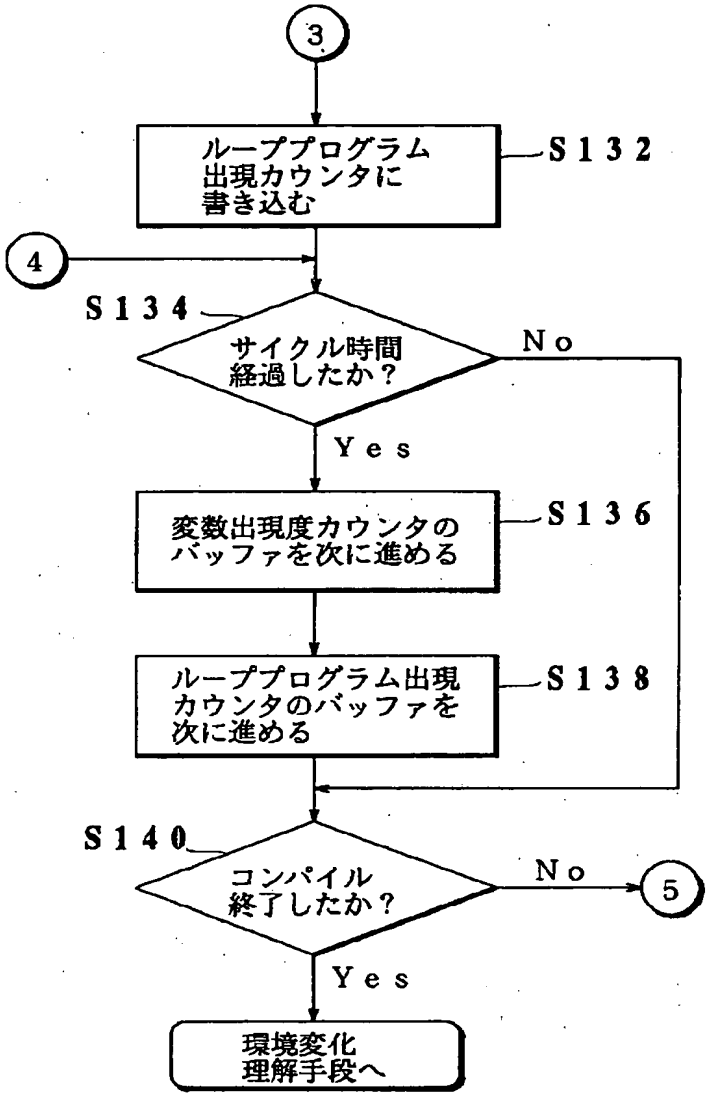
【 図1 1 】


置き換えブロック決定プログラム
各ブロック毎にＬＲＵフラグの重み計数を決定する
S２０
各ブロック毎に使用頻度フラグの重み計数を決定する
S２１
各ブロック毎にＲｅａｄ－Ｗｒｉｔｅフラグの重み計数を決定する
S２２
各ブロック毎の評価値の和を求める
S２３
評価値の最も高いブロックを選定する
S２４
S２５
評価値の最も高いブロックが複数か？
Ｎｏ
Ｙｅｓ
S２６
ＬＲＵフラグの評価値が一番高いブロックを置き換え対象とする
選定されたブロックを置き換え対象とする
S２７
置き換え制御処理へ進む
S２８

【図14】


実行速度情報
コンパイラ評価手段
112
レジスタ割当数
実行形式
108
環境変化理解手段
出現場所情報
制御知識修正手段
制御知識情報
制御知識記憶手段
コンパイラ制御手段
102
カウンタ情報(特徴量)
106
104
環境情報監視手段
110
・変数情報
・ループプログラム情報
・プログラムのソースコード
・CPUのレジスタ数

【 図1 6 】



【 図4 5 】

タスク制御知識

タスク実行先決定用フラグの例

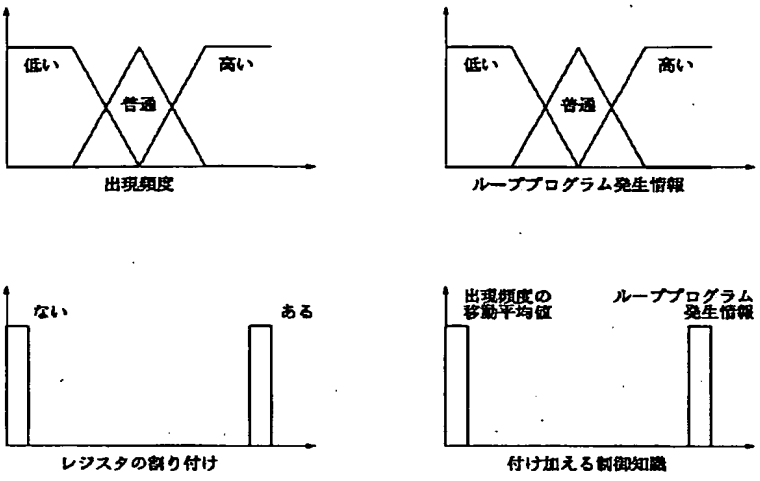
・タスクを分散しても良い場合

処理許可フラグ	すべてのＣＰＵが '1'

・タスクを自分のマシンのＣＰＵで行なう場合

処理許可フラグ	タスク要求が発生したマシンのＣＰＵのフラグは１
処理許可フラグ	タスク要求が発生していないマシンのＣＰＵのフラグは０

【 図2 5 】



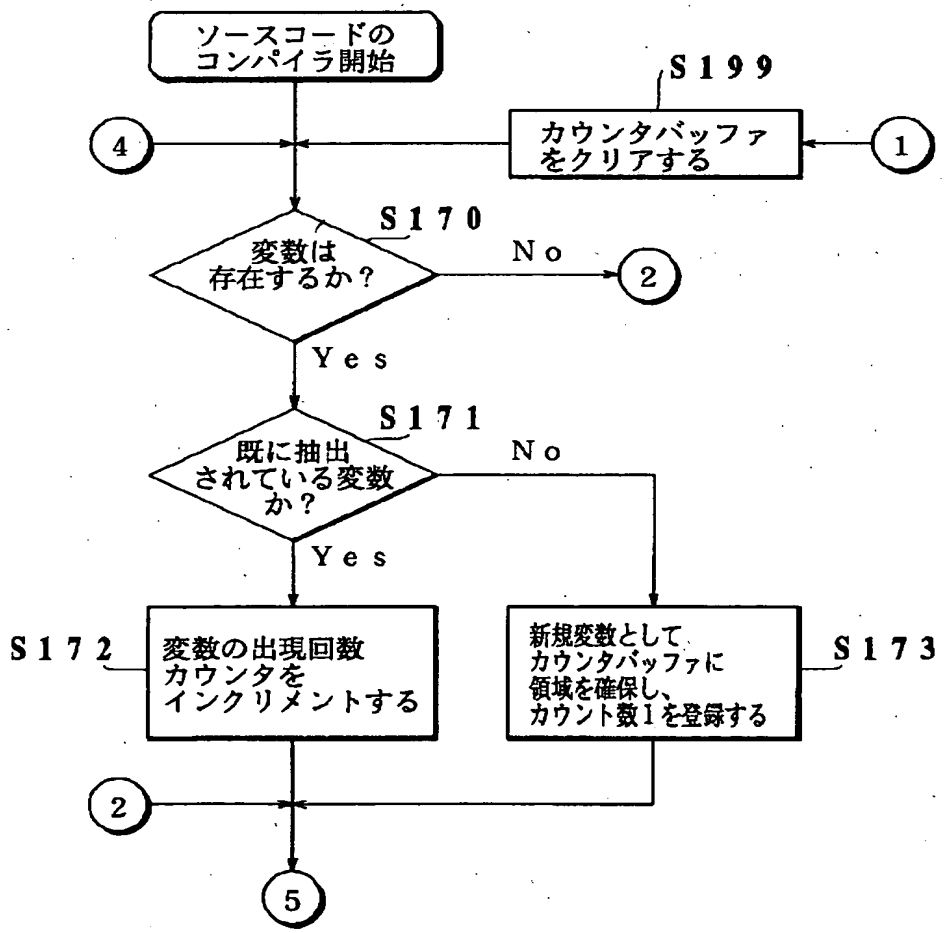
【 図5 1 】

分散コンピューティング環境の混雑度
（ＣＰＵの負荷や、ネットワークの混み具合）

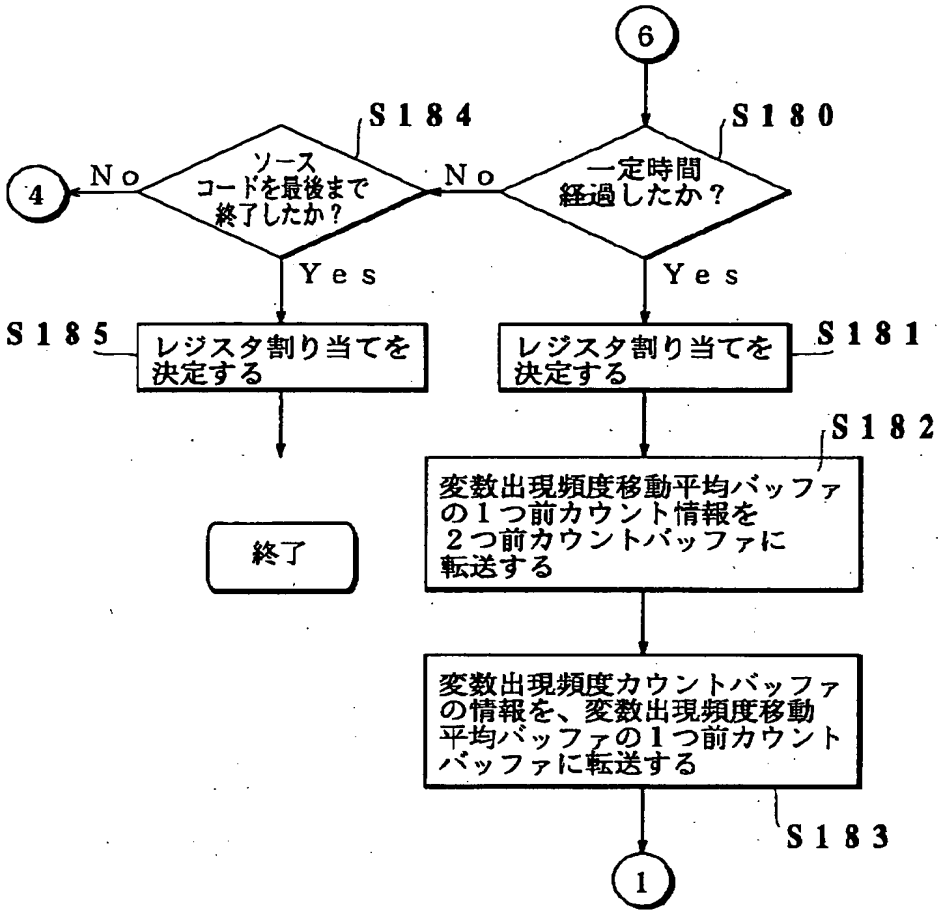
ＩＦ
ＣＰＵ１の負荷が軽い、かつ、ＣＰＵ２の負荷が軽い、かつ、
ＣＰＵ３の負荷が軽い、かつ、全体のタスクが少ない

ＴＨＥＮ
分散コンピューティング環境の混雑度は低い

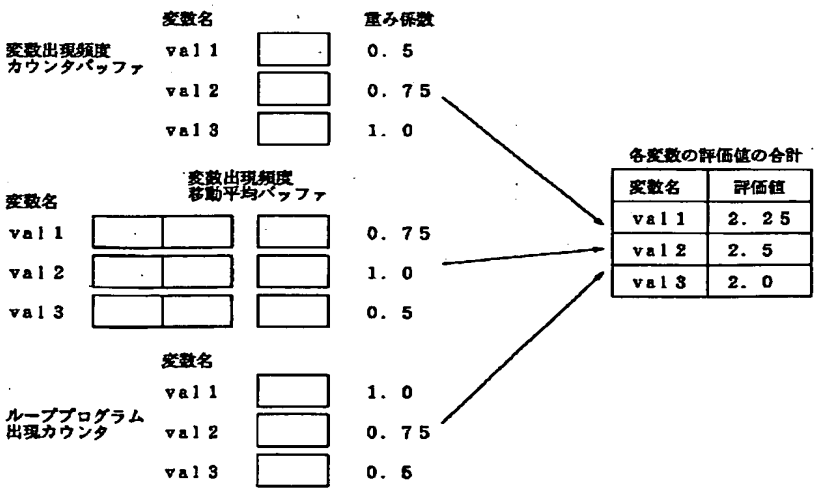
【 図2 7 】



【 図２９ 】



【 図３２ 】



【 図５５ 】

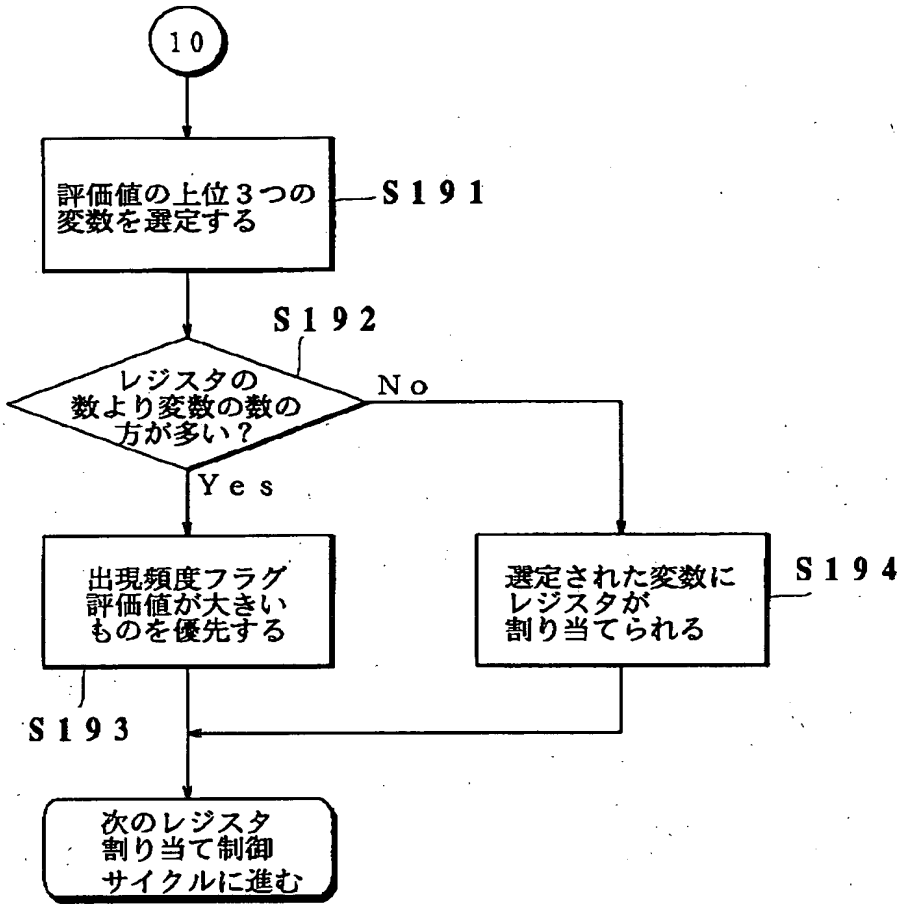
IF
タスクの重要度が高い、かつ、
タスクの優先度が低い

THEN
制御知識に‘使用頻度が多い時には優先度を上げる’知識を
付け加える

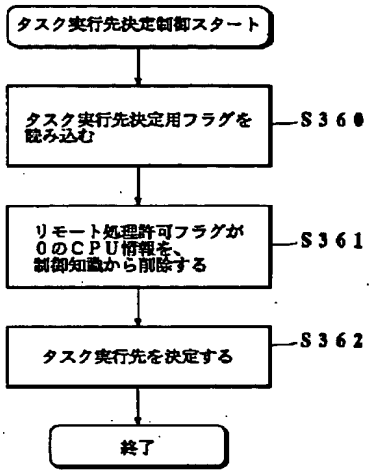
IF
分散コンピューティング環境の混雑度が低い、かつ、
存在するタスクは頻度が高い、かつ、
存在するタスクは重要度が低い

THEN
制御知識を‘タスクを自分のマシンのＣＰＵで行なう’知識に
置き換える

【 図3 1 】

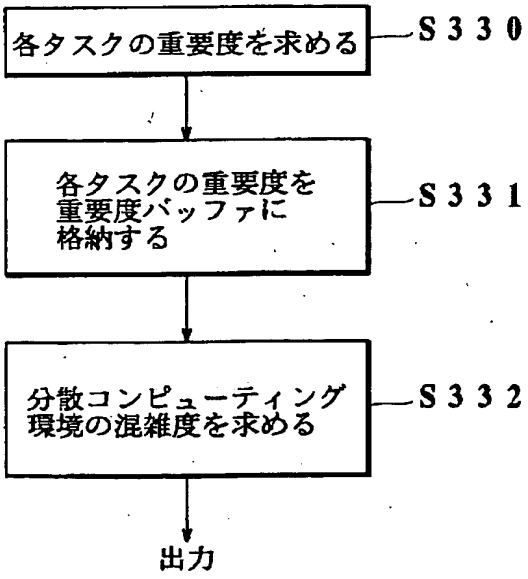


【 図5 9 】

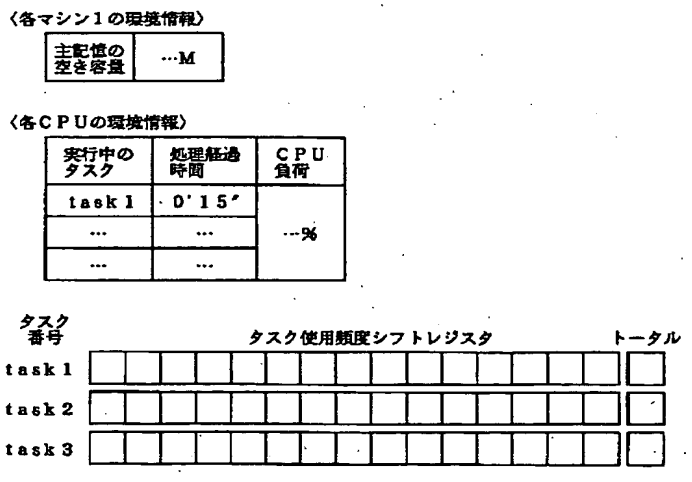


【 図4 1 】

環境変化理解手段のフローチャート

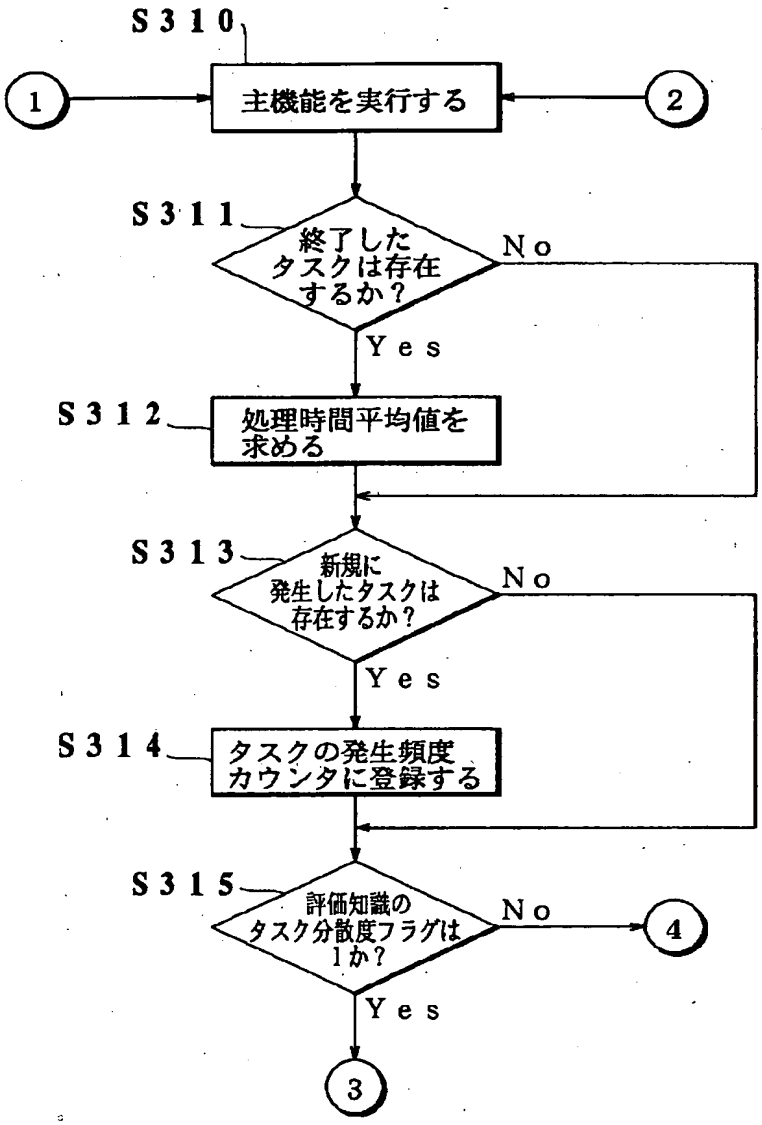


【 図5 7 】

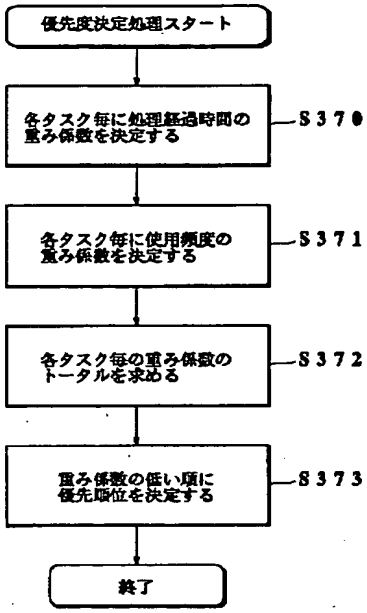


【図３５】

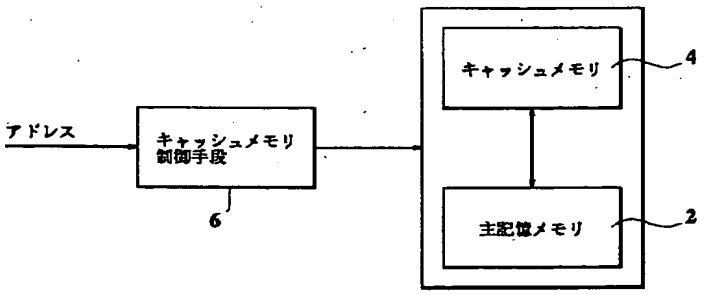
環境情報監視手段のフローチャート（その１）



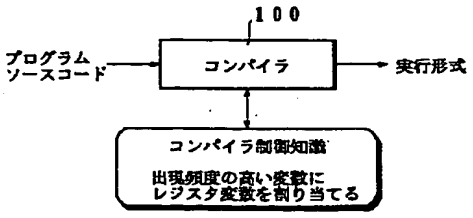
【図６１】



【図６３】

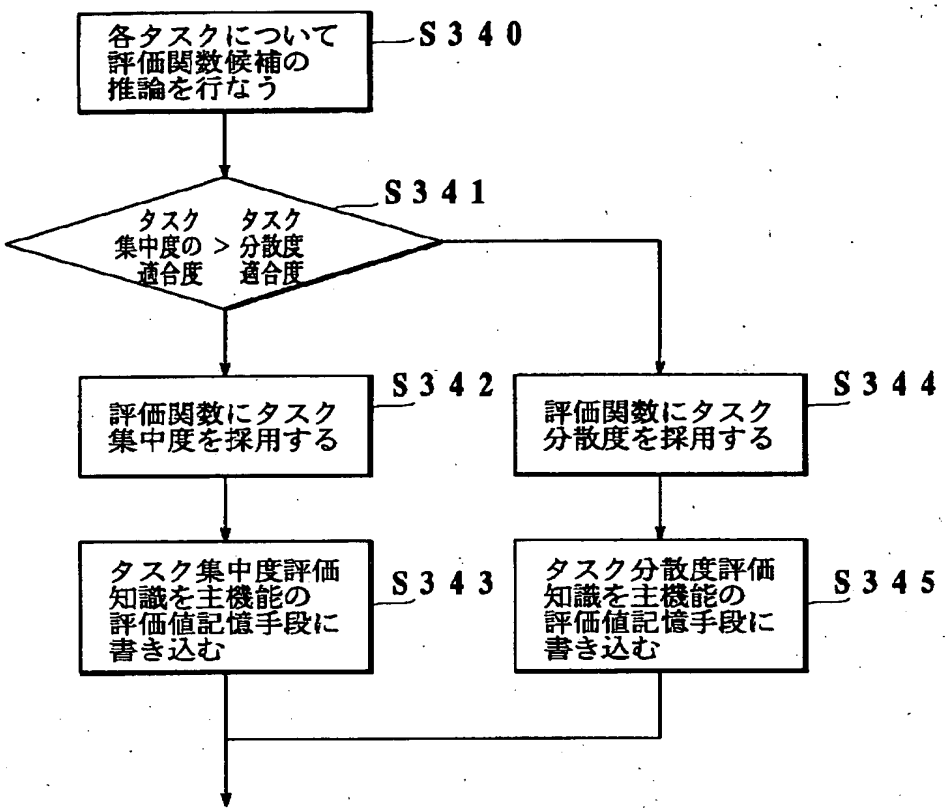


【図６７】



【 図4 2 】

環境情報評価手段のフローチャート



【 図4 7 】

A　発生したタスクの処理時間平均値

task 1	task 2	task 3	task 4	task 5	task 6	task 7
1'30"	0'50"	0'35"	4'12"	3'00"	2'30"	5'00"

B　タスクの発生頻度

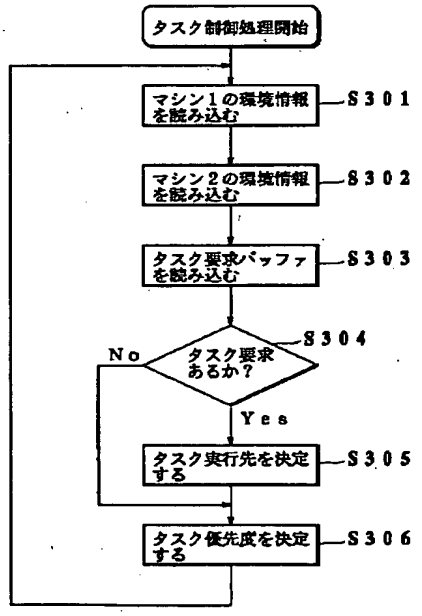
																トータル
task 1	2	2	0	0	0	1	1	3	0	0	0	0	0	2	3	1	15
task 2	1	2	0	0	0	0	1	0	1	1	0	0	0	1	2	0	9
task 7	0	1	7	6	7	1	0	2	1	0	1	0	5	0	0	1	32

タスク　カウント　サイクル

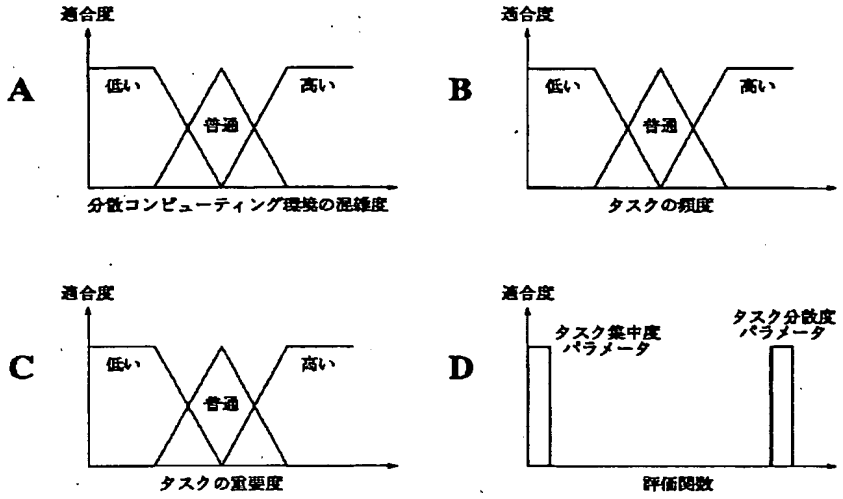
C

	CPU 負荷	主記憶の残り容量
CPU1	70%	1.9M
CPU2	20%	12M
CPU3	40%	6M

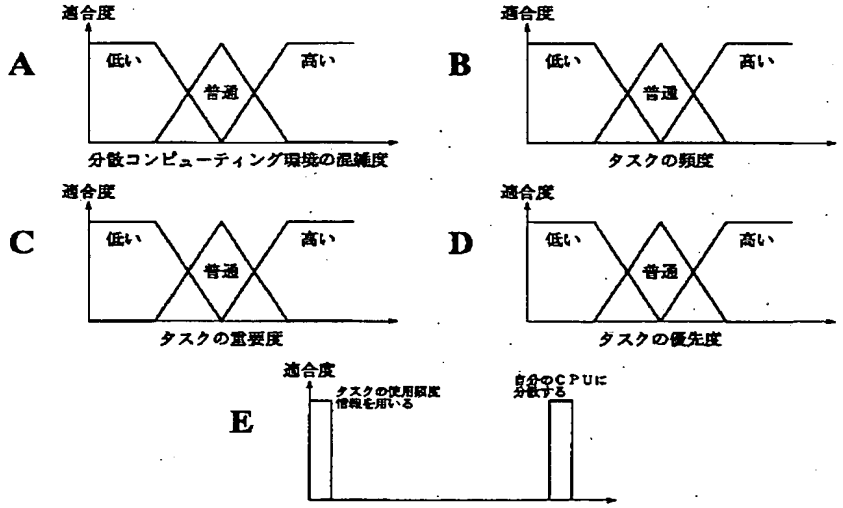
【 図7 2 】



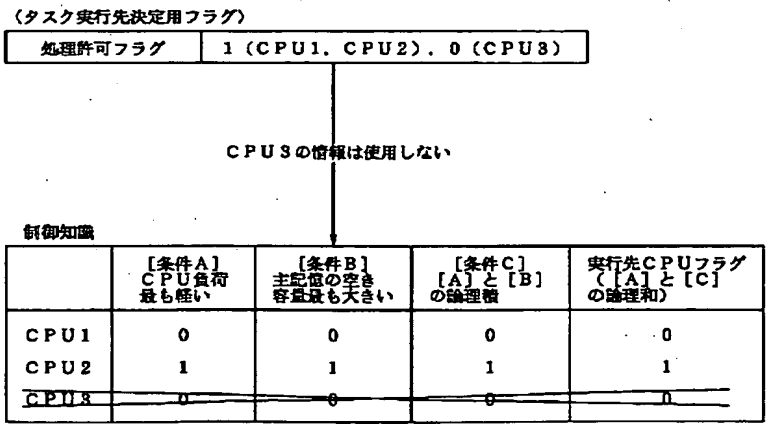
【 図5 4 】


A
適合度
低い
普通
高い
分散コンピューティング環境の混雑度
B
適合度
低い
普通
高い
タスクの頻度
C
適合度
低い
普通
高い
タスクの重要度
D
適合度
タスク集中度パラメータ
タスク分散度パラメータ
評価関数

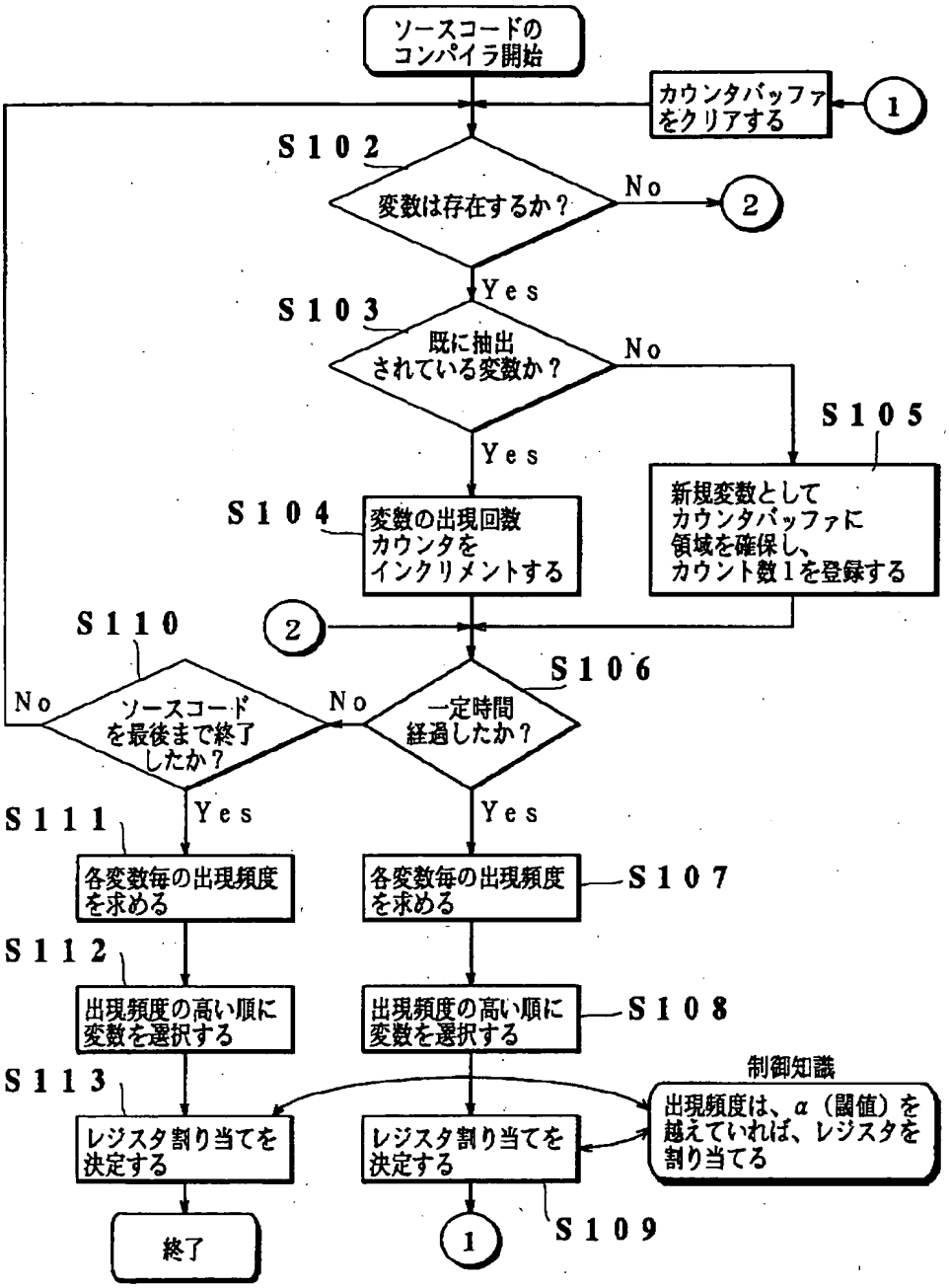
【 図5 6 】


A
適合度
低い
普通
高い
分散コンピューティング環境の混雑度
B
適合度
低い
普通
高い
タスクの頻度
C
適合度
低い
普通
高い
タスクの重要度
D
適合度
低い
普通
高い
タスクの優先度
E
適合度
タスクの使用頻度情報を用いる
自分のCPUに分散する

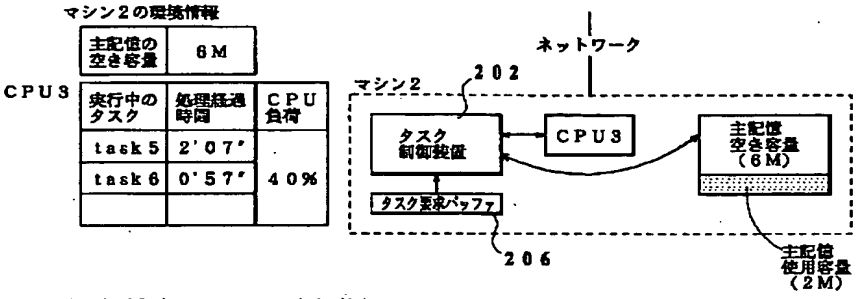
【 図6 0 】


（タスク実行先決定用フラグ）
処理許可フラグ
1（CPU1，CPU2），0（CPU3）
CPU3の情報は使用しない
制御知識
[条件A] CPU負荷最も軽い
[条件B] 主記憶の空き容量最も大きい
[条件C] [A]と[B]の論理積
実行先CPUフラグ（[A]と[C]の論理和）
CPU1 0 0 0 0
CPU2 1 1 1 1
CPU3 0 0 0 0

【図68】


ソースコードのコンパイラ開始
カウンタバッファをクリアする
1
S102
変数は存在するか？
No
2
Yes
S103
既に抽出されている変数か？
No
Yes
S105
新規変数としてカウンタバッファに領域を確保し、カウント数1を登録する
S104
変数の出現回数カウンタをインクリメントする
2
S110
S106
No
ソースコードを最後まで終了したか？
No
一定時間経過したか？
S111
Yes
Yes
各変数毎の出現頻度を求める
各変数毎の出現頻度を求める
S107
S112
出現頻度の高い順に変数を選択する
出現頻度の高い順に変数を選択する
S108
制御知識
S113
出現頻度は、α（閾値）を越えていれば、レジスタを割り当てる
レジスタ割り当てを決定する
レジスタ割り当てを決定する
終了
1
S109

【 図7 1 】



タスク要求バッファ２０６の内容（例）

実行予定タスク	タスク要求発生元	タスク名
	マシン2	task7

【 図7 3 】

タスク実行先決定制御知識

A

	[条件A] CPU負荷最も軽い	[条件B] 主記憶の空き容量最も大きい	[条件C] [A]と[B]の論理積	実行先CPUフラグ（[A]と[C]の論理和）
CPU1	0	0	0	0
CPU2	1	0	0	1
CPU3	0	1	0	0

優先度決定知識…処理時間の長いタスクの優先度をあげる

B

タスク名	処理経過時間	タスク優先度
task1	1'12"	3
task2	0'32"	5
task3	0'15"	6
task4	3'12"	1
task5	2'07"	2
task6	0'57"	4

フロント ページの続き

(51) Int.Cl.[6]　識別記号　庁内整理番号　FI　技術表示箇所
G06F 9/46　340　D　7737－5B
15/163

(72)発明者　田島　年浩
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

(72)発明者　荒尾　真樹
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

(72)発明者　大八木　雅之
京都府京都市右京区花園土堂町10番地　オムロン株式会社内